METAHEURÍSTICA PARA DISMINUIR PENALIZACIONES DEL LAYCAN EN PROGRAMACIÓN DE CARGA DE BUQUES AL GRANEL

Daniel Alfonso Mendoza Casseres
Universidad del Atlántico
danielmendoza@mail.uniatlantico.edu.co

Ronald Andrés Corcho Martínez
Universidad del Atlántico
ron.corcho@gmail.com

Alejandra Berdugo Alonso
Universidad del Atlántico
alejaberdugo0292@gmail.com

(Tipo de Artículo: Investigación. Recibido el 14/11/2014. Aprobado el 04/12/2014)

RESUMEN
La carga de buques al granel es una operación portuaria que se utiliza para transportar cereales, minerales o cargas mixtas. La programación para la carga de los buques, la realiza el fletador teniendo en cuenta una cláusula del contrato de fletamento donde se fija la fecha final de iniciar la carga y la fecha inicial en la cual el buque es requerido (Laycan). Un buque programado fuera del Lyacan causa una penalización monetaria proporcional al tiempo de quebrantamiento. En esta investigación se utiliza una Metaheurística para programar seis buques al granel, los cuales pueden ser cargados simultáneamente por dos *shiploaders* idénticos en un puerto. Se supuso que los buques se programaran por fracciones mediante un *job splitting*. Los resultados obtenidos fueron comparados con la forma habitual de programación, demostrando que la Metaheurística disminuye la penalización total obtenida.

Palabras clave. Metaheurística, Job Splitting, Laycan, Carga al Granel.

METAHEURISTIC FOR REDUCING PENALTIES OF LAYCAN WHEN SCHEDULING BULK SHIP CARGO

ABSTRACT
Bulk ship cargo is a port operation used for transporting cereals, minerals or mixed loads. The charterer schedules ship cargo, considering a clause of the charter agreement where is specified the final date for starting the cargo and the initial date to load on which the ship is required (Laycan). A ship scheduled off of Laycan causes a monetary penalty proportional to the time of brokenness. In this research, a metaheuristic is used for scheduling six bulk ships which can be loaded simultaneously by two identical shiploaders in port. It was supposed that the ships were scheduled fractionally by job splitting. The results were compared with the usual scheduling; demonstrating that metaheuristic decreases the total penalty.

Keywords. *Metaheuristic, Job Splitting, Laycan, Bulk ship*

Métaheuristique pour diminuer les pénalités du laycan dans programmation de chargement des vraquiers

Résumé
Le chargement des vraquiers est une opération portuaire qui est utilisée pour transporter céréales, minéraux ou marchandises mixtes. La programmation pour le chargement des vraquiers est réalisé par le fréteur en considérant une clause du contrat d'affrètement où est fixé la date final de commencer le chargement et la date initiale dans laquelle le vraquier est requis (Laycan). Un vraquier programme dehors du laycan cause une pénalité monétaire qui est proportionnelle au temps d'infraction. On utilise une métaheuristique pour programmer six vraquiers, qui peuvent être chargés simultanément par deux chargeurs de bateau identiques dans un port. On a supposé que les vraquiers sont programmés par fractions au moyen d'un job splitting. Les résultats obtenus ont été comparés avec la manière habituelle de programmation, en démontrant que la métaheuristique diminue la pénalité totale obtenue.

Mots-clés. *Métaheuristique, Job splitting, laycan, vraquier.*

1. INTRODUCCIÓN

Los problemas de secuenciación de trabajos incluyen un conjunto diverso de problemas, los cuales difieren en los parámetros relacionados con la existencia de prioridad, restricciones de precedencia, fecha de vencimiento, entre otras características de los trabajos, si las maquinas son idénticas, diferentes, en paralelo, Job shop, Flow Shop u otras características del entorno de las máquinas o el criterio de optimización tiempo de flujo, máxima demora, total tardanza o tiempo de finalización [1].

Generalmente los problemas de programación de trabajos exime la posibilidad que un trabajo pueda ser procesado simultáneamente en 2 máquinas [2], en el presente artículo el problema será tratado como un Splitting Job pudiendo procesar una misma tarea simultáneamente en máquinas independientes.

La técnica Splitting Job ha sido utilizada para resolver problemas de programación de varias máquinas en paralelo [3], Maquinas con tiempos de alistamiento [4], en sectores como el textil [5], y la metaheurística de algoritmos genéticos ha sido utilizada para resolver problemas de programación de una máquina para minimizar el retraso máximo de los trabajos presentes en la secuencia independientemente de los tiempos de preparación de los grupos de trabajos [6], minimizar la tardanza total en la programación de una maquina [7] y para solucionar la programación de trabajos con penalizaciones [8]. Sin embargo, en una búsqueda preliminar no se encontró aplicación de Splitting Job y/o Algoritmos Genéticos (GA) donde se penaliza el inicio fuera del tiempo programado de un trabajo.

En el presente artículo se considera un problema de secuenciación de buques al granel, el cual tiene las siguientes características: se tienen 2 Shiploaders (maquinas iguales) y 6 Buques (trabajos, todos difieren en capacidad de carga). Cada buque tiene estipulado su Laycan (acuerdo contractual puerto-fletador) y se penalizará su incumplimiento, tiene estimado el tiempo de llegada al puerto el cual se considera cumplirán, su carga inicial es 0.

El puerto solo tiene capacidad para cargar 2 buques simultáneamente, el proceso de cargue es sin interrupción (non-preemptive). En esta investigación se compara la situación actual en la programación de la carga de los buques a granel acorde a la fechas límites (momento final del Laycan) para realizar la carga, con la aplicación de un algoritmo genético para minimizar las penalizaciones por incumplimiento del Laycan.

2. DESARROLLO DEL ARTÍCULO

2.1. Marco referencial

2.1.1. Marco teórico

El problema de secuenciación de tareas o trabajos consiste en especificar el orden en el cual se deben realizar los trabajos acordes al número de máquinas, número de trabajos y las características de ambos factores. Con el objeto de obtener una solución que permita cumplir el criterio de optimización como tiempo de flujo, tiempo de finalización, máxima demora, tardanza total u otro especificado por el analista.

Generalmente, se trata el problema de secuenciación bajo la hipótesis que un trabajo no puede ser procesado por dos máquinas simultáneamente, pero es posible tratar el problema como un Splitting Job donde cualquier parte de un trabajo puede ser procesado en 2 máquinas diferentes al mismo tiempo [2].

El procedimiento de solución de un problema de secuenciación con Splitting Job se puede llevar a cabo con FIFO (Firts In Firts Out), SPT (Shortest Processing Time), EDD (Earliest Due Date), LPT (Lorgest Processing Time), CR (Critical Ratio), S/RO (Slack per Remaining Operations) y Heurística de Johnson; sin embargo, su estructura compleja permite el uso de metaheurísticas llamadas algoritmos evolucionarios (EAs) que generalmente superan métodos convencionales de optimización cuando son aplicados a problemas del mundo real. Los EAs incluyen metaheurísticas de optimización como genetic algorithms (GA), evolutionary programming (EP), evolution strategys (ES), genetic programming (GP), learning classifier systems (LCS), swarm in- telligence (comprising ant colony optimization (ACO) and particle swarm optimization (PSO) [9].

El algoritmo genético ha sido aplicado con éxito en la solución de un problema de programación de trabajos con función de penalización en el cual comparan los resultados del algoritmo genético con los brindados por otros algoritmos, concluyendo que los resultados simulados muestran que el método propuesto es una alternativa viable para la solución de problemas de programación de talleres de trabajo [8].

2.1.2. Marco conceptual

Algoritmo Genético: es un método de búsqueda aleatoria desarrollado a partir de la ley de evolución en el mundo natural. Difiere de las técnicas convencionales, comienza con un conjunto inicial de soluciones aleatorias llamado población la cual satisface las restricciones del entorno o sistema problema. Cada individuo en la población es llamado cromosoma representando una solución al problema en cuestión. Los cromosomas evolucionan a través de cada iteración sucesiva llamada generación, cada cromosoma es evaluado usando medidas de desempeño, los nuevos cromosomas llamados descendientes son formados por fusiones de cromosomas usando operadores de cruce o mutación, luego se seleccionan los mejores cromosomas según las medidas de desempeño y los peores son rechazados pero se mantiene el tamaño de la población constante, los ajustados tienen mayor probabilidad de ser escogidos, luego de varias

iteraciones el algoritmo converge en el mejor cromosoma (solución) [9].

Job Splitting: enfoque en el cual un trabajo puede ser dividido arbitrariamente en partes y ser procesadas independientemente por maquinas diferentes, una parte del trabajo puede ser procesada en dos máquinas diferentes al mismo tiempo [2].

Shiploader: es la máquina de carga de material solido a granel como mineral de hierro, carbón, fertilizantes, granos en los buques para el transporte por mar, es la última pieza en la línea de manejo del material.

Laycan: un plazo de fletamento de naves (Buques en este caso) que representa el número de días permitidos por la autoridad portuaria de un buque para cargar o descargar mercancías sin incurrir en demora o recargos. Especifica la fecha más temprana en la que la estadía puede comenzar y la última fecha en la que puede terminar, después de lo cual el fletador puede optar a cancelar el contrato de fletamento [10].

2.2. Datos del problema para el cargue

A continuación se presentan los datos para una programación desde el 16 al 19 de agosto de 2014.

- Capacidad del Puerto para dos (2) buques.
- Un Shiploaders tiene la capacidad estimada de cargar de 55 a 65 Toneladas por día.
- El número de buques a programar asciende a seis (6).
- La Tabla 1 muestra la carga total a asignar o cargar en cada buque.
- La Tabla 2 expone el Estimated Time to rrival de cada buque.
- La Tabla 3 señala las fechas del Laycan de cada buque.
- La penalización por hora está estipulada en USD 2000.

Tabla 1. Carga total a cargar en cada Buque

Buque	Carga (Ton)
101-102	165000
106-107	170000
112-113	150000
123-124	135000
115-116	170000
104-105	110000

Tabla 2. Estimated Time to arrival

Buque	ETA
101-102	Viernes, 15 de Agosto de 2014
106-107	Viernes, 15 de Agosto de 2014
112-113	Viernes, 15 de Agosto de 2014
123-124	Viernes, 15 de Agosto de 2014
115-116	Viernes, 15 de Agosto de 2014
104-105	Viernes, 15 de Agosto de 2014

Tabla 3. Fechas de Laycan de cada Buque.

Buque	INITIAL LAYCAN	FINAL LAYCAN
101-102	Miércoles, 14 de Agosto de 2014	Sábado, 16 de Agosto de 2014
106-107	Sábado, 15 de Agosto de 2014	Sábado, 16 de Agosto de 2014
112-113	Sábado, 16 de Agosto de 2014	Sábado, 16 de Agosto de 2014
123-124	Domingo, 17 de Agosto de 2014	Martes, 21 de Agosto de 2014
115-116	Miércoles, 19 de Agosto de 2014	Martes, 21 de Agosto de 2014
104-105	Sábado, 18 de Agosto de 2014	Lunes, 18 de Agosto de 2014

2.3. Formulación matemática

2.3.1. Supuestos

- Todos buques Cumplen el Estimated Time to Arrival (ETA).
- La carga inicial de cada Buque será 0.
- La empresa trabaja 3 turnos de 8 horas.
- Cada Shiploader trabaja a una tasa promedio de 60 mil toneladas por día.
- Si la carga de un buque se inicia con Split terminará con Split.
- Non Preemptive.
- El tiempo de alistamiento es 0.
- El tiempo cero será el ETA superior a las 11:59 pm.
- Los trabajos o buques son divididos en 2 sub buques.
- Un Sub buque no podrá esperar la otra parte por más de 16 horas.

2.3.2. Índices

i: número de buque: 1,2,…, n.
j: número de shiploader: 1,2,…, m.
k: sub buque: 1,2,…, p**.**

2.3.3. Parámetros.

SP_{ik}: tiempo del cargue del sub buque k del buque i.
FL_i: tiempo final del Laycan del buque i.
P_i: penalización del buque i por unidad de tiempo.

2.3.4. Variables de Decisión.

T_{ik}: tiempo que inicia el cargue del sub buque k del buque i.
M_{kj}: sub buque k asignado al Ship loader j.

2.3.5. Función Fitness

$$MinZ = \sum_{i=1}^{n} P_i \left\{ \max \left\{ 0, \left[\min \left(T_{i1}, T_{i2}, T_{i3}, ..., T_{im} \right) - F_{Li} \right] \right\} \right\} \quad (1)$$

2.4. Minimización de las penalizaciones por GA

2.4.1. Diagrama para la solución

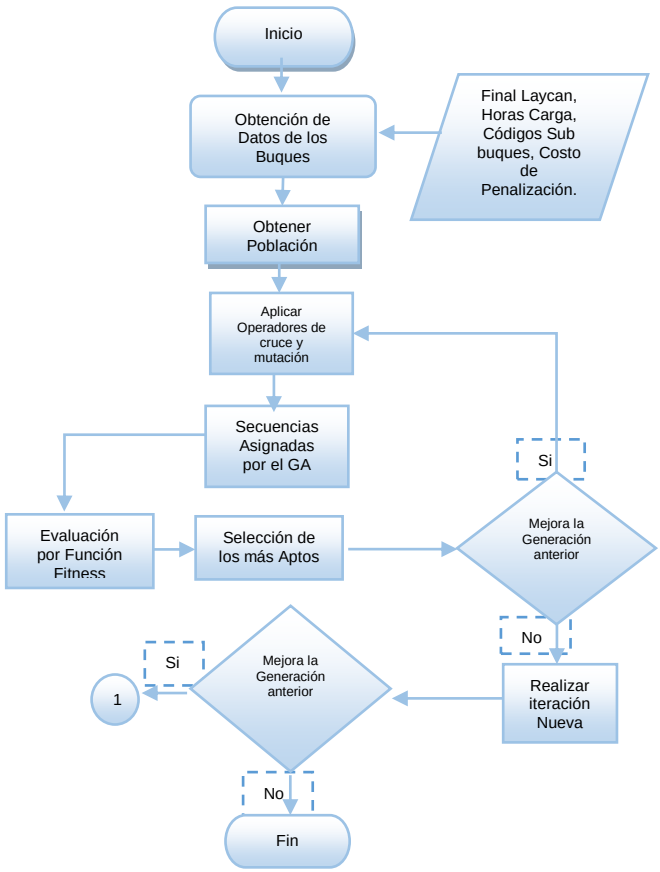


Fig. 1. pasos para solucionar. Adaptado [11].

2.4.2. Procedimiento GA.

En la aplicación de GA se utilizaron números aleatorios del método congruencial lineal, una probabilidad de mutación de 0,3 y probabilidad de cruce de 0,7, longitud de los cromosomas de 16 Genes en código binario con una extensión de 7 números, la población inicial con tamaño de dos Cromosomas.

El primer cromosoma fue hallado acorde a la situación actual de programación en el puerto, que está en función de la fecha límite del Laycan, y el segundo acorde a la fecha inicial del Laycan. Luego se realizan las respectivas generaciones aplicando los pasos del algoritmo genético y el código adaptado que se menciona a continuación.

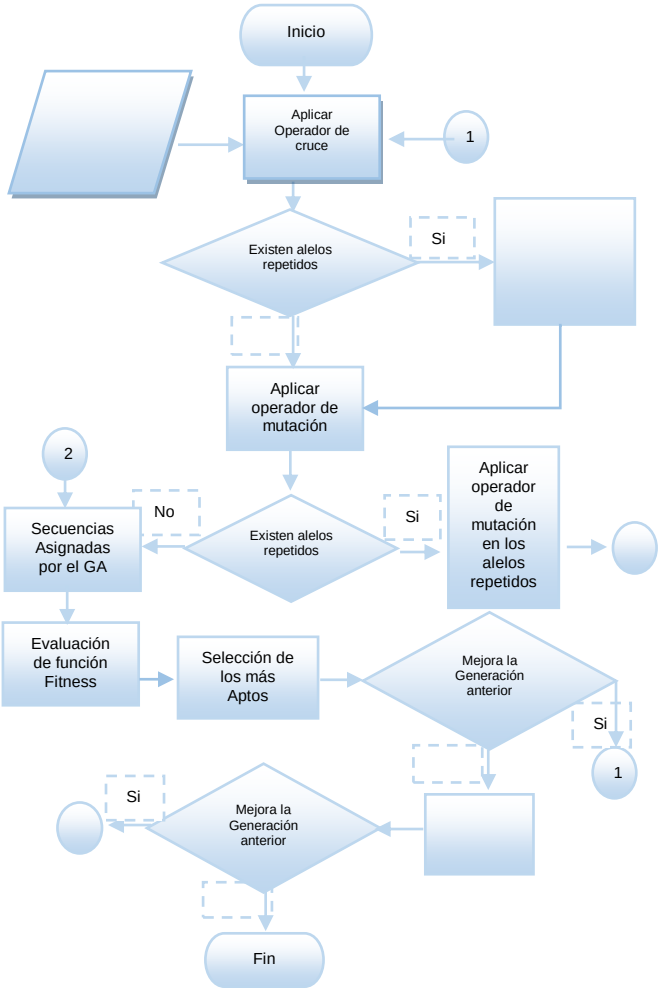
Tabla 4. Representación Binaria de los sub buques

Buque	Código Binario
101	1100101
102	1100110
106	1101010
107	1101011
112	1110000
113	1110001
123	1111011
124	1111100
115	1110011
116	1110100
104	1101000
105	1101001

Tabla 5. Cromosoma inicial 1. Programación en función a la fecha límite del Laycan

101	1100101
102	1100110
106	1101010
107	1101011
112	1110000
113	1110001
104	1101000
105	1101001
123	1111011
124	1111100
115	1110011
116	1110100

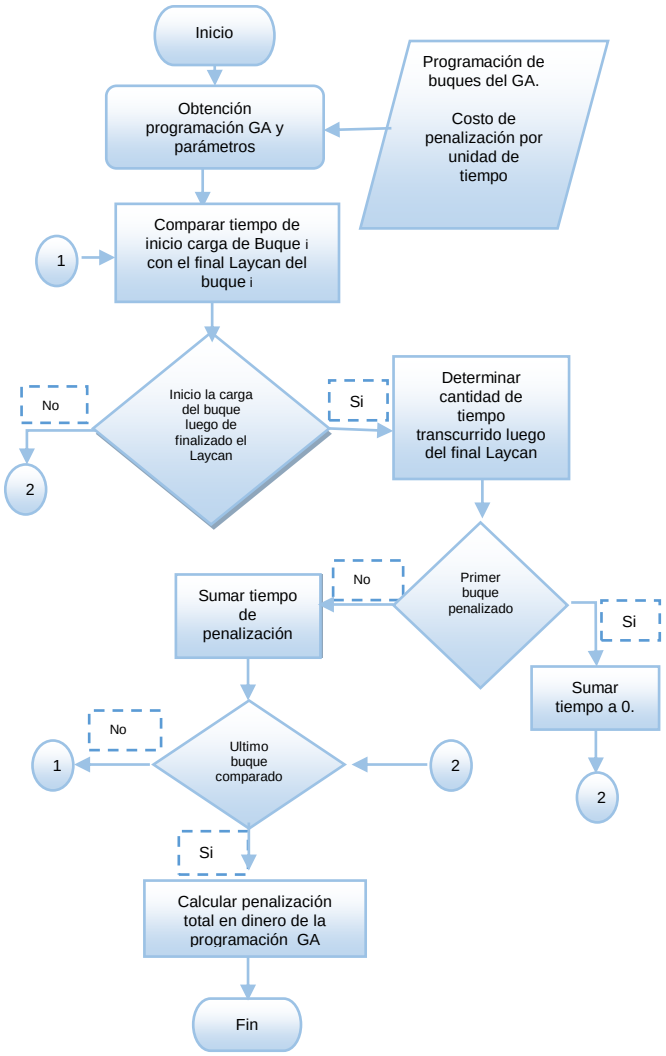
Código GA adaptado [12]
Proceso: GA.
Entrada: Datos del problema, Población inicial y parámetros del GA.
Salida: La mejor solución.



2.4.3. Algoritmo para la solución con java

Presentación de parte de la estructura del código de programación en el programa JAVA, en el cual se determina las penalizaciones en las programaciones establecidas por el GA:

Proceso: Determinación de penalización.
Entradas: Programación establecida por el GA y costo de penalización por unidad de tiempo.
Salida: Penalización total de la programación.



2.5. Resultados

A continuación se presentan la situación actual en el problema y los resultados obtenidos con la aplicación del GA.

Tabla 6. Situación actual de la programación de los buques.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
1	1	0	66	
6	2	0	68	
2	1	67	126	84.000
4	2	69	112	
3	2	113	166	
5	1	127	194	
Total Penalización				84.000

Los Individuos de la población fueron obtenidos a través de la aplicación de heurística basada en el Final Laycan más cercano y el Estimated Time to Arrival.

Tabla 7. Programación sub buques acorde al final Laycan más cercano.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
107	2	34	67	
112	1	68	97	86.000
113	2	68	97	
104	1	98	119	50.000
105	2	98	119	
123	1	120	146	
124	2	120	146	
115	1	147	180	4.000
116	2	147	180	
Total Penalización				158.000

Tabla 8. Programación acorde al Estimated Time to Arrival.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
107	2	34	67	
112	1	68	97	86.000
113	2	68	97	
115	1	98	131	
116	2	98	131	
104	1	132	153	118.000
105	2	132	153	
123	1	154	180	18.000
124	2	154	180	
Total Penalización				240.000

Generación 1 del algoritmo genético, iteración en la cual se cruzó y mutó a los individuos de la población inicial.

Tabla 9. Programación del Individuo 1, de la primera generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
112	2	34	63	18.000
113	2	64	93	

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
107	1	68	101	
115	2	93	127	
116	1	102	135	
124	2	128	154	
123	1	136	162	
105	2	155	176	164.000
		Tabla 9		
104	1	163	184	
Total Penalización				200.000

Tabla 10. Programación del Individuo 2, de la primera generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
107	2	34	67	
112	1	68	97	
113	2	68	97	86.000
104	1	98	119	
116	2	98	131	50.000
115	1	120	153	
105	2	132	153	
123	1	154	180	
124	2	154	180	18.000
Total Penalización				172.000

Generación 2 del algoritmo genético, iteración en la cual se cruzó y mutó, al individuo 1 de la población inicial y al individuo 2 de la primera generación.

Tabla 11. Programación del Individuo 1, de la segunda generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
107	2	34	67	
123	1	68	94	
113	2	68	97	86.000
112	1	95	124	
124	2	98	124	
116	1	125	158	
115	2	125	158	
104	1	159	180	172.000
105	2	159	180	
Total Penalización				276.000

Tabla 12. Programación del Individuo 2, de la segunda generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
112	2	0	30	
113	2	31	60	
102	1	34	66	
106	2	61	94	72.000
107	1	67	100	
115	2	95	128	
104	1	101	122	58.000
105	1	123	144	
116	2	129	162	
123	1	145	171	
124	2	163	189	
Total Penalización				130.000

Generación 3 del algoritmo genético, iteración en la cual se cruzó y mutó, el individuo 1 de la población inicial y el individuo 2 de la segunda generación.

Tabla 13. Programación del Individuo 1, de la tercera generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
106	2	0	34	
102	1	34	66	84.000
107	2	35	68	
112	1	67	96	
113	2	69	98	
123	1	97	123	
104	2	99	120	52.000
105	2	121	142	
124	1	124	150	
115	2	143	176	
116	1	151	184	
Total Penalización				136.000

Tabla 14. Programación del Individuo 2, de la tercera generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
107	2	34	67	
112	1	68	97	86.000

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
113	2	68	97	
115	2	98	131	
105	1	98	119	50.000
104	1	120	141	
116	2	132	165	
123	1	142	168	
124	1	169	195	
Total Penalización				154.000

Generación 4 del algoritmo genético, iteración en la cual se cruzó y mutó, el individuo 2 de la segunda generación y el individuo 1 de la tercera generación.

Tabla 15. Programación del Individuo 1, de la cuarta generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	33	
106	1	34	67	18.000
112	2	34	63	18.000
113	2	64	93	
107	1	68	101	
104	2	94	115	42.000
105	1	102	123	
124	2	116	142	
123	1	124	150	
115	2	143	176	
116	1	151	184	
Total Penalización				78.000

Tabla 16. Programación del Individuo 2, de la cuarta generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
106	2	0	34	
102	1	34	66	
107	2	35	68	
112	1	67	96	84.000
113	2	69	98	
104	1	97	118	48.000
105	2	99	120	
123	1	119	145	
115	2	121	154	
116	1	146	179	
124	2	155	181	
Total Penalización				132.000

Generación 5 del algoritmo genético, iteración en la cual se cruzó y mutó, el individuo 2 de la segunda generación y el individuo 1 de la cuarta generación.

Tabla 17. Programación del Individuo 1, de la quinta generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
102	2	0	34	
106	1	34	67	18.000
113	2	35	67	18.000
112	2	68	93	
107	1	68	101	
115	2	94	127	
104	1	102	123	58.000
105	1	124	145	
116	2	128	161	
123	1	146	171	2.000
124	2	162	188	
Total Penalización			96.000	

Tabla 18. Programación del Individuo 2, de la quinta generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
101	1	0	33	
112	2	0	30	
113	2	31	60	
102	1	34	66	
106	2	61	94	72.000
107	1	67	100	
104	2	95	116	44.000
105	1	101	122	
123	2	117	143	
124	1	123	149	
115	2	144	177	
116	1	150	183	
Total Penalización				116.000

Generación 6 del algoritmo genético, iteración en la cual se cruzó y mutó, el individuo 1 de la cuarta generación y el individuo 1 de la quinta generación.

Tabla 19. Programación del Individuo 1, de la sexta generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
106	1	0	34	
112	2	0	30	
113	2		60	

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
107	1		68	
101	2		93	72.000
102	1		101	
115	2		127	
104	1		123	58.000
105	1		145	
116	2		161	
123	1		172	2.000
124	2		188	
Total Penalización				132.000

Tabla 20. Programación del Individuo 2, de la sexta generación.

ID ARRIVAL (SHIP)	No. Ship loader	Tiempo inicio de carga (hora)	Tiempo en que libera Ship loader (hora)	Penal. ship USD
112	1	0	30	
113	2	0	30	
101	1	31	63	12.000
106	2	31	64	12.000
102	1	64	96	
107	2	65	98	
104	1	97	118	48.000
105	2	99	120	
123	1	119	145	
124	2	121	147	
115	1	146	179	
116	2	148	181	
Total Penalización			72.000	

El algoritmo converge en el individuo 2 de la sexta generación.

2.6. Análisis de los resultados

Los resultados en las Tablas 6-20 muestran que la solución inicial donde se programan los buques de acuerdo al Final Laycan más cercano, conllevó a una penalización con costos de USD 84.000. El puerto fue penalizado por un buque cuando se realizó la carga por dos Shiploaders.

La programación donde se supone que todos los buques cumplen el Estimated Time to Arrival (ETA) como también que la carga de un buque puede tener Split, permitió una menor penalización. El puerto fue penalizado por 3 buques cuando se realizó la carga por dos Shiploaders, pero con una penalización de USD 72.000.

Los resultados muestran que programar la carga de los buques con Shiploaders idénticos como un Job Splitting con una implementación con Algoritmos Genéticos (GA), arroja mejores resultados que una programación de buques ordenándolos de acuerdo a las fechas límites del Laycan.

La programación con algoritmos genéticos (GA) conllevo a que algunos buques con tiempos más cercanos a los límites del Laycan fueron programados de primero, pero no necesariamente conservarán el orden de acuerdo a la fecha de finalización del Laycan.

Los cromosomas que se desarrollaron a través de iteraciones sucesivas mostraron que las generaciones cruzadas y mutadas mejoraban la evaluación en la Función Fitness de manera eficiente.

3. TRABAJOS FUTUROS

Se propone relajar algunos supuestos como trabajar con variabilidad en la velocidad de carga de los Shiploader, introducir el tiempo alistamiento en el proceso de carga de buques al granel o variabilidad en los tiempos del Estimated Time to Arrival para mejorar la calidad en aplicaciones prácticas.

4. CONCLUSIONES

El problema de programación en puertos que permita cumplir con una cláusula del contrato de fletamento llamada Laycan, con Shiploaders idénticos pudo analizarse como un problema con Splitting implementando algoritmos genéticos (GA).

La utilización de la herramienta propuesta resulta efectiva para la programación de buques en puertos al granel, generando un mejor ordenamiento de los buques para realizar la carga en comparación a la forma actual como se realiza en el puerto carbonífero, donde se ordenan de acuerdo a las fechas límites del Laycan. Contribuyendo en la solución del problema de programación, minimizando costos por incumplimientos del Laycan por parte del puerto.

La GA fue una herramienta adecuada en la programación de buques en puertos al granel con los supuestos planteados; sin embargo, se hace necesario seguir trabajando para relajar los supuestos y tener una cercanía mejor a los problemas de programación en puertos.

5. AGRADECIMIENTOS

Los autores agradecen a una empresa del sector portuario que se dedica a realizar carga al granel, por aceptar la realización de esta investigación.

6. REFERENCIAS

[1] V. Gabrel. Scheduling jobs within time windows on identical parallel machines: New model and algorithms. Elsevier Science B.V. European Journal of Operational Research, Vol. 83, issue 2, pp 320-329. Jun. 1995.

[2] W. Xing and J. Zhang. Parallel machine scheduling with splitting jobs. Elsevier Science B.V. Discrete Applied Mathematics, Vol. 103, issues 1-3, pp. 259–269, Jul. 2000.

[3] J. L. Hurink; W. Kern and W. Nawijn. Scheduling split-jobs on parallel machines. Twente, University of Twente, Faculty of Mathematical Sciences. NL-7500 AE Enschede. Jun. 2000.

[4] F. Schalekamp; R. Sitters; S. Van der Ster; L. Stougie; V. Verdugo and A. Van Zuylen. Split scheduling with uniform setup times. Journal of scheduling. 10.1007/s10951-014-0370-4. Jan. 2014.

[5] P. Serafini. Scheduling jobs on several machines with the job splitting property, Udine, University of Udine, Department of Mathematics and Computer Science. Jan. 1996.

[6] H.NAZIF and L. S. LEE. Solving Single Machine Scheduling Problem with Maximum Lateness Using a Genetic Algorithm. Journal of Mathematics Research, Vol. 2, Number 3, p. 5, Aug. 2010.

[7] G. A. Süer; X. Yang; O. I. Alhawari; J. Santos and R. Vazquez. A Genetic Algorithm Approach for Minimizing Total Tardiness in Single Machine Scheduling. International Journal of Industrial Engineering and Management (IJIEM), Vol. 3, issue 3, pp. 163-171, Sep. 2012.

[8] L. Sun; X. Cheng and Y. Liang. Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function. International Journal of Intelligent Information Processing, Vol. 1, Number 2, Dec. 2010.

[9] M. Gen; R. Cheng and L. Lin. Network Models and Optimization Multiobjective Genetic Algorithm Approach. Springer-Verlag, London, 2008, p. 1.

[10] Dutch port guide. (2013, Jan.) glossary, Laycan. [Online]. Available:http://www.dutchportguide.com/component/option,com_glossary/id,525/.

[11] M. Gen; R. Cheng and L. Lin. Network Models and Optimization Multiobjective Genetic Algorithm Approach. Springer-Verlag, London, 2008, p. 2.

[12] M. Gen; R. Cheng and L. Lin. Network Models and Optimization Multiobjective Genetic Algorithm Approach. Springer-Verlag, London, 2008, p. 3.